LA-UR-21-27190
Approved for public release; distribution is unlimited.

Title: OpenSesame tutorial

Author(s): Rehn, Daniel Adam
Greeff, Carl William
Sheppard, Daniel Glen
Starrett, Charles Edward
Crockett, Scott

Intended for: Report

Issued: 2021-07-22

OpenSesame tutorial

Daniel A. Rehn[1], Carl W. Greeff[2], Daniel G. Sheppard[1], Charles E. Starrett[1], and Scott D. Crockett[2]

[1]Computational Physics Division, Los Alamos National Laboratory
[2]Theoretical Division, Los Alamos National Laboratory

April, 2021

Abstract

OpenSesame is a program for generating tabular equations of state (EOS), with capabilities for multiphase EOS construction. In this tutorial, we provide an overview of how to run OpenSesame to construct a multiphase EOS. We discuss some general features of OpenSesame, followed by a description of sample input files required for multiphase EOS construction. We also discuss how to extract data from EOS tables in order to compare to experimental data, with an example using the OpenSesame GUI. Lastly, we provide a description of how to generate ASCII-formatted EOS tables most often used by hydro code users.

Overview

OpenSesame [1] is a program for generating tabular equations of state (EOS) with support for the construction of multiphase EOS [2]. Multiphase EOS are the main focus of this tutorial.

The basic thermodynamic variable used in OpenSesame is the Helmholtz free energy, denoted F and sometimes A. OpenSesame relies on a decomposition of the total free energy into 3 pieces:

$$F(V,T) = E_c(V) + F_{\text{ion}}(V,T) + F_{\text{el}}(V,T). \tag{1}$$

These terms are, from left to right, the *cold curve* or static lattice energy, the free energy associated with ionic motion, and the electronic free energy. Each term is computed using a different materials model, and multiple choices of materials models are available for each term. The choice of materials models are up to the user and depend upon the material phase (solid, liquid, gas, etc.), as well as the user's particular preferences in their choice of models. In each case, the models are evaluated over a range of V and T and stored in tables (tabular format). Those tables can then be interpolated to provide data, such as isotherms, Hugoniots, etc., that can be compared to experiments.

Once $F(V,T)$ is computed, other thermodynamic quantities can be evaluated using $F(V,T)$. Some examples include the following [3],

$$E = F + TS = F - T\left(\frac{\partial F}{\partial T}\right)_V \qquad \text{internal energy}$$

$$S = -\left(\frac{\partial F}{\partial T}\right)_V \qquad \text{entropy}$$

$$P = \rho^2\left(\frac{\partial F}{\partial \rho}\right)_T \qquad \text{pressure}$$

$$C_V = -T\left(\frac{\partial^2 F}{\partial T^2}\right)_V = \left(\frac{\partial E}{\partial T}\right)_V = T\left(\frac{\partial S}{\partial T}\right)_V \qquad \text{specific heat at constant volume}$$

$$C_P = \left(\frac{\partial H}{\partial T}\right)_P = T\left(\frac{\partial S}{\partial T}\right)_P \qquad \text{specific heat at constant pressure}$$

$$B_T = V\left(\frac{\partial^2 F}{\partial V^2}\right)_T = -V\left(\frac{\partial P}{\partial V}\right)_T \qquad \text{isothermal bulk modulus}$$

$$\alpha = -\frac{1}{V}\left(\frac{\partial V}{\partial T}\right)_P \qquad \text{thermal expansion coefficient}$$

$$\Gamma = V\left(\frac{\partial P}{\partial E}\right)_V \qquad \text{Grüneisen parameter}$$

Running OpenSesame involves three main steps, shown schematically in Fig. 1. Typically, these steps are repeated in a cyclical manner so that model parameters can be adjusted to provide good agreement with experimental data. In the following sections, we discuss each of these 3 steps in detail.

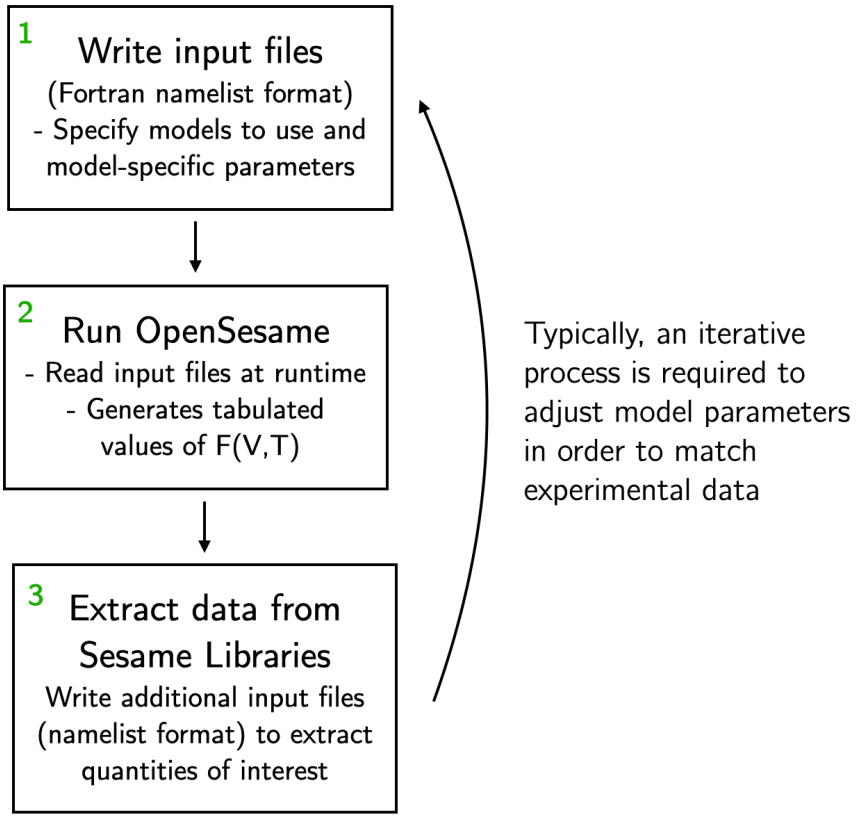


Figure 1: Three main steps in running OpenSesame

Unless otherwise noted, units used in OpenSesame are those in Table 1.

Variable	Description	Units
ρ (R)	density	g/cm^3
V	specific volume	cm^3/g
T	temperature	K
P	pressure	GPa
E, F, A, G	energy (Internal, Helmholtz, Gibbs)	MJ/kg
C_V, C_P	specific heat	MJ/kg/K
S	entropy	MJ/kg/K
B_T, B_S	bulk modulus	GPa
u, c_s	velocity, sound speed	km/s
α_v (alpha)	volume thermal expansion coefficient	1/K
μ (mu), G (muM)	shear modulus at T=0, shear modulus at melt	GPa

Table 1: Variable names and units used in OpenSesame

Below, we will use SESAME 2161 (material = tin) as an example. SESAME 2161 was developed by Carl Greeff [4] and is a multiphase EOS with 2 solid phases (labeled β and γ) and the liquid phase. The phase diagram of SESAME 2161 is shown in Fig. 2.

Note that a new tin EOS, SESAME 2162, has also been developed and includes 4 solid phases of tin [5, 6], and therefore can be thought of as the successor to 2161. However, for tutorial purposes, it is easier to discuss the simpler case of just 2 solid phases.

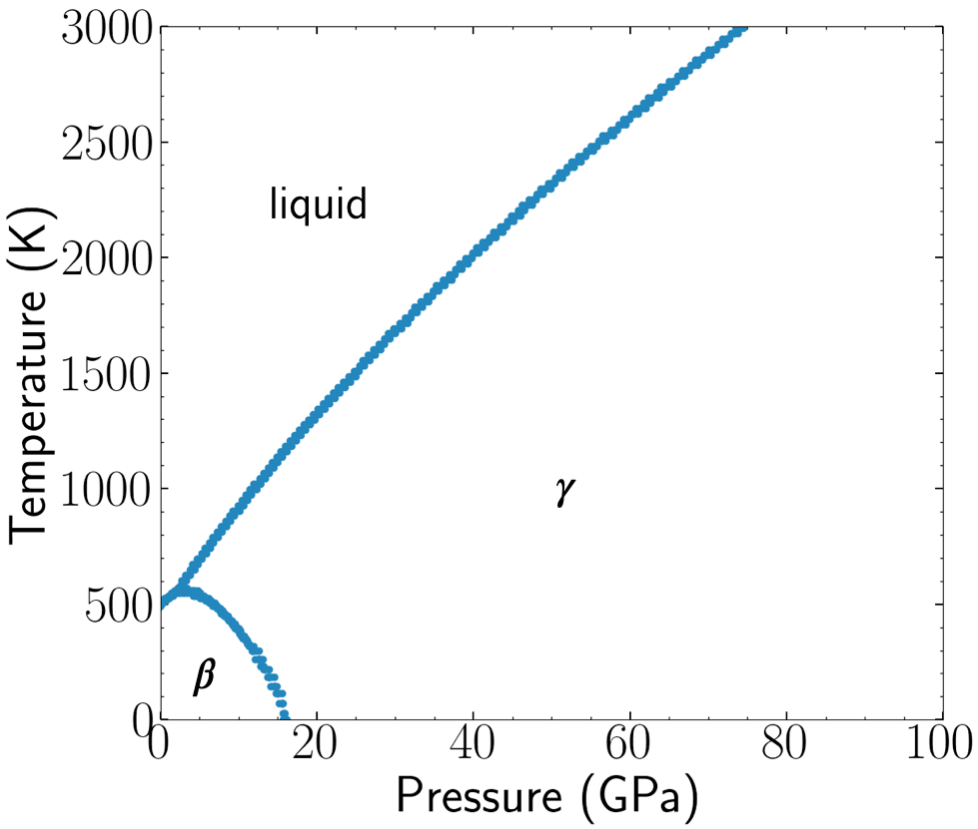


Figure 2: Phase diagram of tin from SESAME 2161. β and γ are the solid phases

1. Writing OpenSesame input files

OpenSesame relies on input files in Fortran namelist format [7]. Although it is possible to write just one input file for a full multiphase EOS, it is sometimes preferable to split the input files into parts that correspond to the individual phases. Here, we show a typical directory structure for a multiphase EOS,

```
$ ls
1-beta/    2-gamma/    3-liquid/    4-multiphase/    lib/    plots/    run.sh
```

Folders `1-3` contain input files for the individual phase tables, while `4-multiphase` contains the input file for the multiphase construction. The `lib` folder contains the tables produced by running the input files in folders `1-4` and `plots` includes plotting scripts, discussed later. Lastly, `run.sh` is a bash script that runs OpenSesame in each folder.

The contents of these folders is as follows,

```
$ ls *
1-beta:
input.nml

2-gamma:
input.nml

3-liquid:
input.nml

4-multiphase:
input.nml

lib:
002101/    002102/    002103/    002104/    002161/    material_directory
```

The folders `1-4` contain only an `input.nml` file. After running `opensesame`, other ASCII format output files are generated in these directories. The `lib` directory contains folders that store tabulated thermodynamic quantities. The numbers (`002101`, `002102`, etc.) of these directories correspond to each phase, and the numbers themselves are specified in the `input.nml` files. More details on `lib` are provided later in the tutorial.

β phase input

The input file for the β phase in `1-beta` is copied in Listing 1. Note the comments in green font. The cold curve model is the Rose-Vinet model [8], the ionic model is the Debye model [9], and the electronic model is the Thomas-Fermi-Dirac model [10, 11, 12].

```
&job
    job_type = 'neweos'          ! must select job type, create new EOS
    resultlib_path = '../lib'
/
&neweos                          ! define parameters for new EOS
    material_number      = 2101 ! specify mat # (corresponds to lib/002101)

    ! basic material properties
    material_name        = 'tin'
    author               = 'C. Greeff & E. Chisolm, T-1'
    atomic_number        = 50
    atomic_weight        = 118.710
    reference_density    = 7.28729
    ifstandard           = .false.  ! nuanced options, see manual
    match_low            = .false.

    ! cold curve parameters
    cold_model           = 'rose'
    cold_density         = 7.42402
    cold_bulk_modulus    = 57.0
    cold_dbdp            = 4.9

    ! ion model parameters
    nuclear_model        = 'debye'
    gruneisen_option     = 7
    reference_debye      = 144.306
    reference_gamma      = 2.2
    dgamma_left          = -2.2
    dgamma_right         = -2.2

    ! electronic model parameters
    electron_model       = 'tfd'
    elec_low_temp_interp = .true.
/
```

Listing 1: `1-beta/input.nml`: input file for β phase.

γ phase input

The input file for the γ phase in `2-gamma` is shown in Listing 2. Again, note the descriptive comments in green font. The models used for γ can be the same as those of the β phase, but here `hightliq` is used for some complicated reasons related to the re-emergence of γ phase in the liquid region.

```
&job
    job_type = 'neweos'
    resultlib_path = '../lib'
/

&neweos
    material_number          = 2102  ! corresponds to lib/002102

    ! basic material properties
    material_name            = 'tin'
    author                   = 'C. Greeff & E. Chisolm, T-1'
    atomic_number            = 50
    atomic_weight            = 118.710
    reference_density        = 7.02426
    ifstandard               = .false.
    match_low                = .false.

    ! cold curve parameters
    cold_model               = 'rose'
    cold_density             = 7.51329
    cold_bulk_modulus        = 50.0
    cold_dbdp                = 5.6
    energy_shift             = 0.0387499

    ! ion model parameters
    nuclear_model            = 'hightliq'  ! or 'debye'
    gruneisen_option         = 7
    reference_debye          = 90.7148
    reference_gamma          = 3.2
    dgamma_left              = -4.48
    dgamma_right             = -4.48

    ! electronic model parameters
    electron_model           = 'tfd'
    elec_low_temp_interp     = .true.

    ! melt parameters if using hightliq
    melt_option              = 'lindemann'
    initial_melt_density     = 7.595
    initial_melt_temperature = 650.0
    liquid_shift             = .true.
    entropy_difference       = 0.0
/
```

Listing 2: `2-gamma/input.nml`: input file for the γ phase.

Liquid phase input

The input file for the liquid phase in `3-liquid` is shown in Listing 3. Again, note the descriptive comments in green. The ionic model used here is described in Refs. [13, 14].

```
&job job_type='neweos'
    resultlib_path='../lib'
/

&neweos
    material_number          = 2103  ! corresponds to lib/002103

    ! basic material properties
    material_name            = 'tin'
    author                   = 'C. Greeff & E. Chisolm, T-1'
    atomic_number            = 50
    atomic_weight            = 118.710
    ifstandard               = .false.

    ! cold curve parameters
    cold_model               = 'rose'
    cold_density             = 7.51329
    cold_bulk_modulus        = 50.0
    cold_dbdp                = 5.6
    energy_shift             = 0.0387499

    ! ion model parameters
    nuclear_model            = 'hightliq'
    gruneisen_option         = 7
    reference_debye          = 90.7148
    reference_gamma          = 3.2
    dgamma_left              = -4.48
    dgamma_right             = -4.48

    ! electronic model parameters
    electron_model           = 'tfd'
    elec_low_temp_interp     = .true.

    ! melt parameters
    melt_option              = 'lindemann'
    initial_melt_density     = 7.595
    initial_melt_temperature = 650.0
    liquid_shift             = .true.
    entropy_difference       = 0.5412

    ! additional parameters related to behavior in expansion
    reference_density        = 7.02426
    cohesive_energy          = 2.3904
    lennard_jones_exponent   = 0.9
/
```

Listing 3: `3-liquid/input.nml`: input file for the liquid phase.

Multiphase input

The multiphase file is long, so we break it into 2 main parts:

1. Grid construction
2. Multiphase construction

Grid construction

The grid consists of two parts: a compression (ρ/ρ_0) grid and temperature (T) grid. The compression grid is defined in Listing 4.

```
&job job_type = 'grid' /

&grid
    grid_type   = 'compression'
    grid_change = 'new'  ! create a new compression grid
    grid_size   = 85
    grid_new    = 0.0000e+00, 0.1000e-05, 0.2000e-05, 0.5000e-05, 0.1000e-04,
                  ...
                  0.8000e+02, 0.9000e+02, 0.1000e+03, 0.1250e+03, 0.1500e+03
/

&job job_type = 'grid' /

&grid
    grid_type     = 'compression'
    grid_change   = 'insert'   ! insert a grid on the existing compression
   grid
    !grid_file     = ''  ! option to import a grid from a file
    grid_spec     = 'lin'
    grid_size     = 420
    grid_limit_lo = 0.9600e+00
    grid_limit_hi = 0.2100e+01
/
```

Listing 4: `4-multiphase/input.nml`: defining the compression grid

Note that the first grid is specified with `grid_change='new'`, generating a new compression grid and the values are defined manually with different compression values (some lines are left out for brevity, denoted by `...`). After this, the second grid specification is of type `'insert'`, which adds a linearly spaced grid of 420 points between 0.96 and 2.1. This option is mostly used to provide a refined grid of points over particularly relevant compression ranges.

After defining the compression grid, we define the temperature grid in a similar way, shown in Listing 5.

```
&job job_type               = 'grid' /
&grid
    grid_type               = 'temperature'
    grid_change             = 'new'  ! construct a new grid
    grid_temperature_units  = 'ev'
    grid_file               = ''
    grid_size               = 54
    grid_new                =
      0.0000e+00, 0.2500e-02, 0.6250e-02, 0.1000e-01, 0.1250e-01,
      ...
      0.8000e+03, 0.9000e+03, 0.1000e+04, 0.1500e+04, 0.2500e+04,
/

&job job_type               = 'grid' /

&grid
    grid_type               = 'temperature'
    grid_change             = 'insert'   ! add to existing grid
    grid_temperature_units  = 'ev'
    grid_size               = 25
    grid_spec               = 'lin'
    grid_limit_lo           = 0.2750e-01
    grid_limit_hi           = 0.5000e-01
/

!  some additional 'insert' grids are omitted for brevity here
```

Listing 5: `4-multiphase/input.nml`: defining the temperature grid

Note here that we again use a `'new'` grid and an `'insert'` grid to refine it, with additional `'insert'` grids left out for brevity. Temperatures can be specified in eV or K.

Multiphase construction

After defining the grids, we turn to the multiphase construction. The multiphase construction is broken into two parts: an initial "non-standardized" table called 2104, followed by a "standardized" table called 2161. The standardization step is described below.

The primary multiphase construction input is shown in Listing 6.

```
&job
  job_type = 'materials'
  sourcelib_path = '../lib'
  resultlib_path = '../lib'
/

&materials
    author              = 'C. Greeff & E. Chisolm, T-1'
    references          = 'LA-UR-05-9414'
    material_option     = 'multiphase'
    result_material     = 2104
    reference_density   = 7.28729

    ! specify material ids for 2104 creation
    nmats               = 3
    source_materials(1) = 2101
    source_materials(2) = 2102
    source_materials(3) = 2103
    phase_names(1)      = 'beta'
    phase_names(2)      = 'gamma'
    phase_names(3)      = 'liquid'

    ! windows to limit range of given phases
    ph_rholo(1)         = 6.0
    ph_rhohi(1)         = 10.0
    ph_thi(1)           = 1000.0
    ph_rholo(2)         = 7.0
/
```

Listing 6: `4-multiphase/input.nml`: construction of the non-standardized 2104 table

The 2104 multiphase table is built from the 3 phase tables described above, with numbers `2101,2102,2013`. Also note the last four lines: these variables specify "windows" that determine the min and max values of density and temperature for each phase. The point of the windows is to prevent the reappearance of certain phases in regions of the phase diagram where they don't belong. For example, here `ph_rholo(2)` says that the minimum density of the gamma phase (material index 2 defined above) is 7 g/cm^3. Therefore, in the multiphase construction, gamma phase cannot reappear at densities below 7 g/cm^3.

After the definition of the multiphase 2104 table, we create the 2161 EOS via a "standardization" step, described above. The standardization step simply resets the zero of energy to a value at room T (298.15 K) and ambient pressure. This is done because hydro codes prefer to define the 0 of energy to ambient conditions. The input lines to do this are as in Listing 7.

```
!  Standardization step
&job job_type           = 'materials'
    sourcelib_path      = '../lib'
    resultlib_path      = '../lib'
/

&materials
    author              = 'Eric Chisolm, T-1'
    material_option     = 'standardize'
    source_materials    = 2104
    result_material     = 2161
/
```

Listing 7: `4-multiphase/input.nml`: standardization step to create the 2161 table.

After standardization the 2161 table itself could be considered "complete". However, depending on the needs of the users, other options for additional tables to add to the 2161 EOS are available. These include,

1. Maxwell construction
2. Construction of melt curve tables
3. Generation of shear modulus tables

These do not necessarily have to be done. Here we focus only on melt curve table generation, which is frequently desired by hydro code users.

The input for melt curve generation is shown in Listing 8.

```
!  Generating melt curves
&job
    job_type       = 'neweos'
    sourcelib_path = '../lib'
    resultlib_path = '../lib'
/

&neweos
    material_number               = 2161
    material_name                 = 'tin'
    author                        = 'C. Greeff & E. Chisolm, T-1'
    atomic_number                 = 50
    atomic_weight                 = 118.710
    reference_density             = 7.28729
    melt_gamma_reference_density  = 7.02426
    melt_model                    = 'multiphase'
    melt_option                   = 'lindemann'
    melt_gruneisen_option         = 7
    melt_reference_debye          = 90.7148
    melt_reference_gamma          = 3.2
    melt_dgamma_left              = -4.48
    melt_dgamma_right             = -4.48
    initial_melt_density          = 7.595
    initial_melt_temperature      = 650.0
    solid_multiplier              = 0.99
    liquid_multiplier             = 1.00
/
```

Listing 8: `4-multiphase/input.nml`: generation of melt curve tables

Parameters for the melt table are basically the same as the liquid phase, but notice we specify some `melt_` variable options. These behave in the same way as the liquid parameters, it is just that here the point is to generate melt curves in a different format of (T, P) points or (ρ, T) points.

2. Running OpenSesame

I recommend installing OpenSesame on a personal laptop because it runs much faster than on the HPC machines and the data storage is then local. After building the source code, you can add the location of the binaries to your `PATH` variable in a `/.bashrc` (or analogous) file. At that point, the following binaries should be available:

```
/path/to/OpenSesame/OpenSesameSource/opensesame
/path/to/OpenSesame/OpenSesameSource/gui/opensesamegui.real
```

The first of these can be run in each directory where an `input.nml` file is located, for example via,

```
$ opensesame < input.nml
```

The GUI executable requires slightly more work to set up. For this, you will need to create an additional executable file located at

```
/path/to/OpenSesame/OpenSesameSource/gui/opensesamegui
```

The contents of this file consist of just one line:

```
echo "source /path/to/OpenSesame/OpenSesameSource/gui/opensesamegui.real ; main" | wish
```

Then you will want to allow executable permissions on that file:

```
$ chmod u+x /path/to/OpenSesame/OpenSesameSource/gui/opensesamegui
```

It is then possible to run `opensesamegui` from the command line. We will show examples of using the GUI to plot different quantities in Part 3 below.

`run.sh` script

Recall the directory structure from above, where we see a `run.sh` script,

```
$ ls
 1-beta/   2-gamma/   3-liquid/   4-multiphase/   lib/   plots/   run.sh
```

The contents of `run.sh` are shown in Listing 9.

```
#!/bin/bash
SESAME=opensesame
# create lib folder for output
[ ! -d './lib' ] && mkdir lib
{
  folder='1-beta'
  echo '-> running in' $folder;
  ti=`date +%s`;
  cd $folder;
  $SESAME < input.nml > output
  tf=`date +%s`;
  echo '-> ' $folder ' runtime (seconds) = ' $((tf-ti))
} &
wait;  # optionally take this out to run 1-beta, 2-gamma simultaneously
{
  folder='2-gamma'
  echo '-> running in' $folder;
  ti=`date +%s`;
  cd $folder;
  $SESAME < input.nml > output
  tf=`date +%s`;
  echo '-> ' $folder ' runtime (seconds) = ' $((tf-ti))
} &
wait;  # optionally take this out to run 2-gamma, 3-liquid simultaneously
{
  folder='3-liquid'
  echo '-> running in' $folder;
  ti=`date +%s`;
  cd $folder;
  $SESAME < input.nml > output
  tf=`date +%s`;
  echo '-> ' $folder ' runtime (seconds) = ' $((tf-ti))
} &
wait;  # do not remove: multiphase must be done after all 3 phases finish
{
  folder='4-multiphase'
  echo '-> running in' $folder;
  ti=`date +%s`;
  cd $folder;
  $SESAME < input.nml > output
  tf=`date +%s`;
  echo '-> ' $folder ' runtime (seconds) = ' $((tf-ti))
} &
wait;
```

Listing 9: `run.sh` script

Note that the `wait` statements after the β and γ cases can in principle be removed to allow the β, γ and liquid phase construction to occur in parallel. However, the multiphase construction must be done after the completion of the individual phases, since it relies on those tables already being present to form the 2104 table.

3. Extract data from Sesame libraries

We now describe how to extract data from the OpenSesame output so that comparisons to relevant experiments can be made. There are two main parts to this section,

1. A description of the types of data contained in the `lib` folder
2. How to run opensesame to extract data from `lib` and compare to experimental data

Step 2 can be done by hand, using `opensesame` directly, or it can be done using the `opensesamegui`. We will look at the `opensesamegui` way first, since this method automatically generates opensesame input files that can be then be used to plot data.

Layout of the `lib` directory

As shown earlier, the `lib` folder consists of the following,

```
$ ls lib
002101/   002102/   002103/   002104/   002161/   material_directory
```

Each of the numbered directories contains subdirectories. The `2161` table is of most interest, so we will look at its subdirectories. Note that other individual phase tables are laid out similarly. The subdirectories of `2161` are,

```
$ ls lib/002161
101     103     311/     401/     412/     432/
102     301/    321/     411/     431/     table_directory
```

Here we see both files and folders labeled by different numbers. The numbers have a specific meaning that are listed in `table_directory`, with a brief description here [3, 15]:

- 100 series (comments)
  - 101: provides basic information: name, authors, etc.
  - 102-199: free-form text describing an other information
- 201: atomic number, atomic weight, reference density, etc.
- 300 series (thermodynamic functions on a ρ, T grid)
  - 301: total ρ, T, P, E, F
  - 303: cold + nuclear only
  - 304: electronic only
  - 305: ionic only
  - 306: cold curve ρ, T, P, E, F
  - 311: Maxwell constructed 301 table
  - 321: mass fractions for multiphase EOS (used to plot phase diagram)

- 400 series (functions along a curve)
  - 401: vapor dome ρ, T, P, E, F
  - 411: solidus ρ, T, P, E, F
  - 412: liquidus ρ, T, P, E, F
  - 431: shear modulus at $T = 0$
  - 432: shear modulus at $T = 0$ and $T = T_{\text{melt}}$
- 500 series (opacities)
- 600 series (conductivity)

If we now look inside, for example the `301` (total) folder, we see the following:

```
$  ls 002161/301/
A      E      P       R       T       item_directory
```

In short, each of these are files that contain different thermodynamic quantities on the defined grids. Basic information about these files, e.g., their sizes and number of points, etc. are found in the `item_directory` file.

Extract data from `lib` to compare with experiments

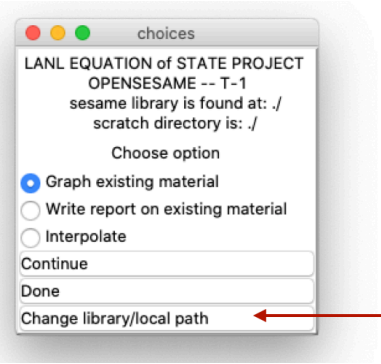
The tables themselves should not be used directly. Instead, we want to generate additional input files that run OpenSesame to extract useful quantities from these tables. In this way, OpenSesame does all of the interpolation and data handling for us. The easiest way to extract data is by running the OpenSesame GUI. Running the GUI automatically creates OpenSesame input files that can subsequently be used to generate data for plotting.

To demonstrate use of the GUI, we return to the root directory of EOS we are working on and create a new folder called `gui`, then we move to that directory and execute `opensesamegui`,

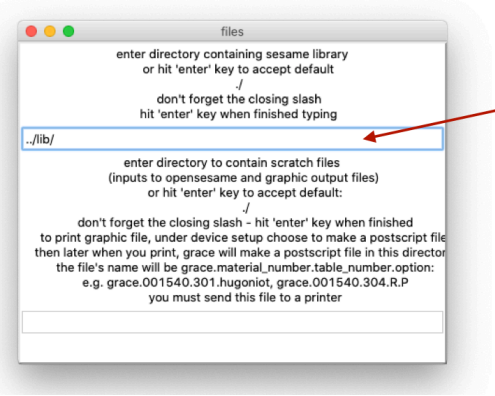
```
$ ls
1-beta/    2-gamma/    3-liquid/    4-multiphase/    lib/    plots/    run.sh
$ mkdir gui
$ cd gui
$ opensesamegui
```

After running this, a GUI window should pop up. It may require X Windows (on Mac, XQuartz) to be installed. The following pages will be completely graphical, showing how to use the GUI.

Running the GUI



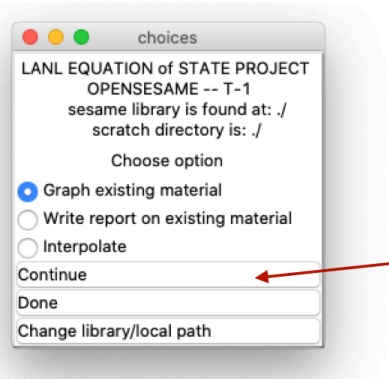
click here: we first need to change the default library path



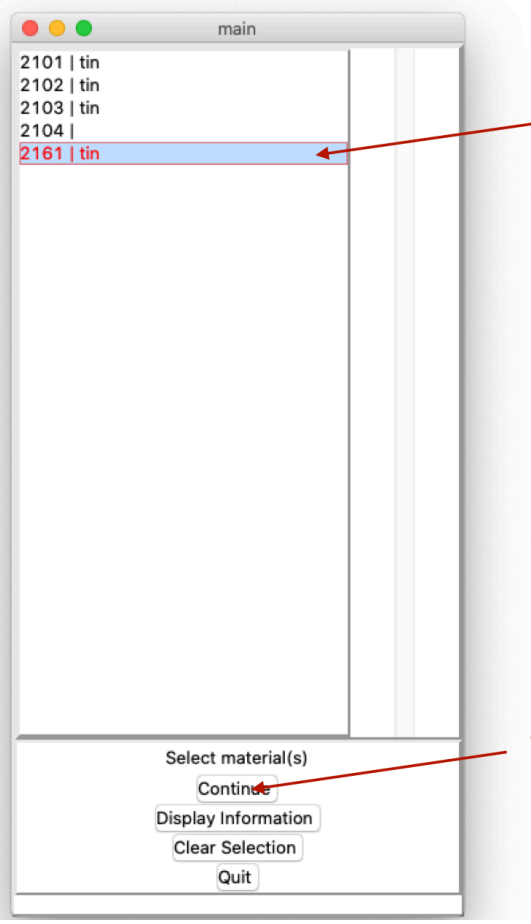
specify the path to the library, which is now ../lib/

NOTE: the / at the end MUST be included or else the GUI will crash!

After doing this, hit TAB and then ENTER until this window closes

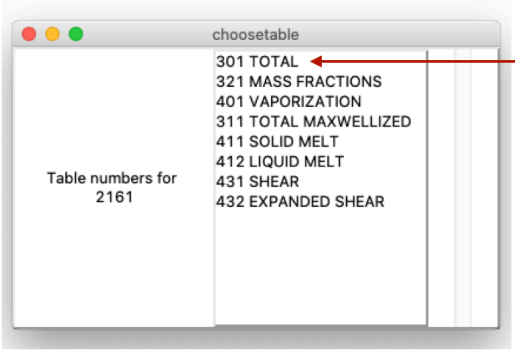


Return to this window, having changed the library path and now click Continue to graph existing material

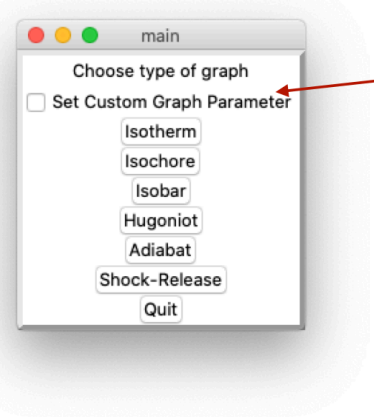


Highlight the material you want to graph. In this case we will look at the multiphase 2161 table

Then click continue

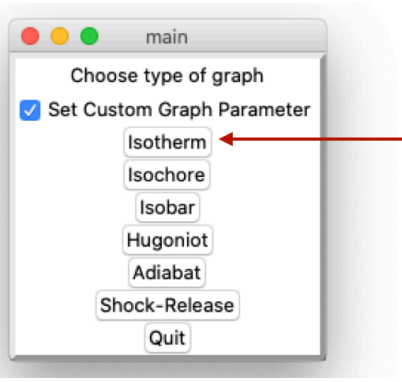


We will look at quantities from the total 301 table

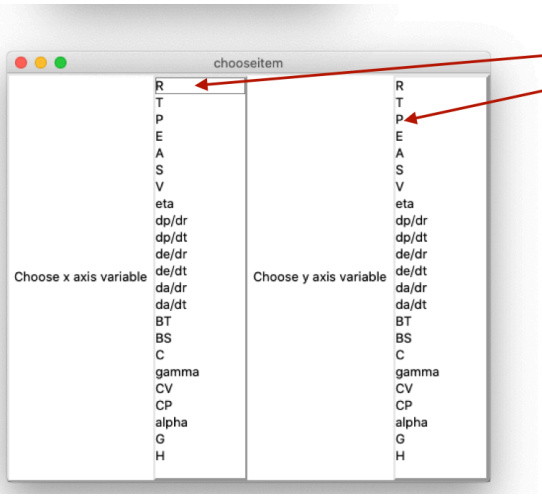


This window now pops up, giving us the option to plot different types of quantities.

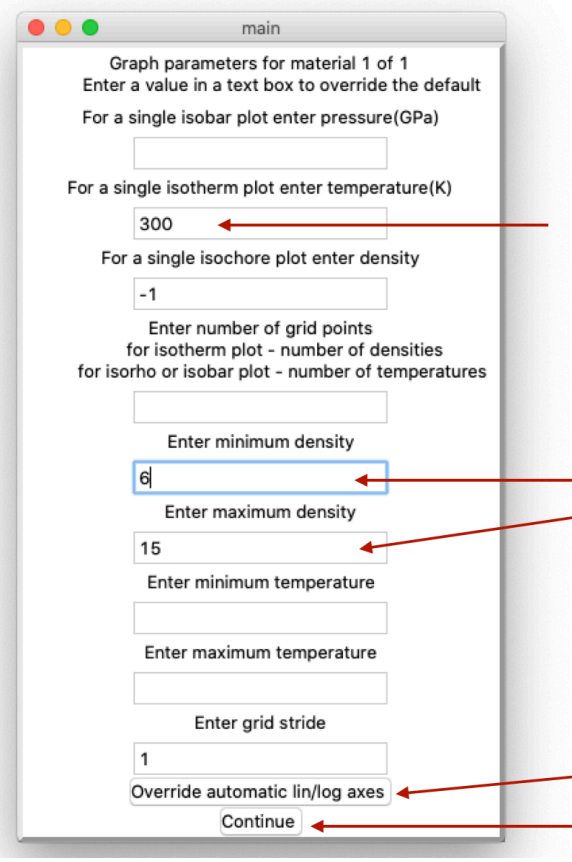
Note that it is VERY helpful to check the "Set Custom Graph Parameters" option. Without selecting this, all isotherms, isochores, etc. are plotted simultaneously on logarithmic axes, which is of limited usefulness for comparing to experimental data



As an example, we will plot a room temperature isotherm



An option for the x and y axes comes up. We will look at P(ρ), so we select these two quantities

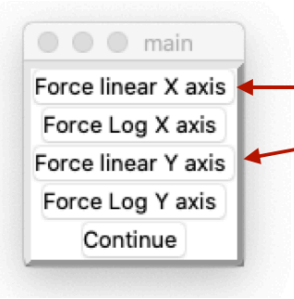


We will plot the room T isotherm (300 K)

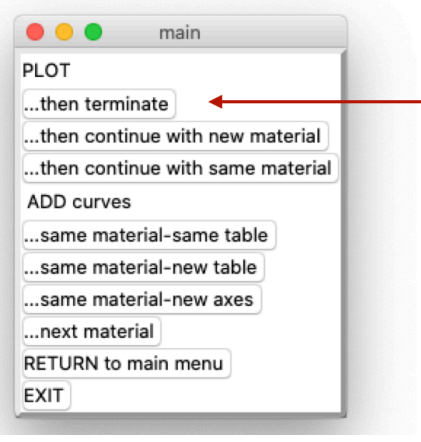
Limit to a density range of 6 to 15 g/cc

Option to select linear vs. logarithmic axes

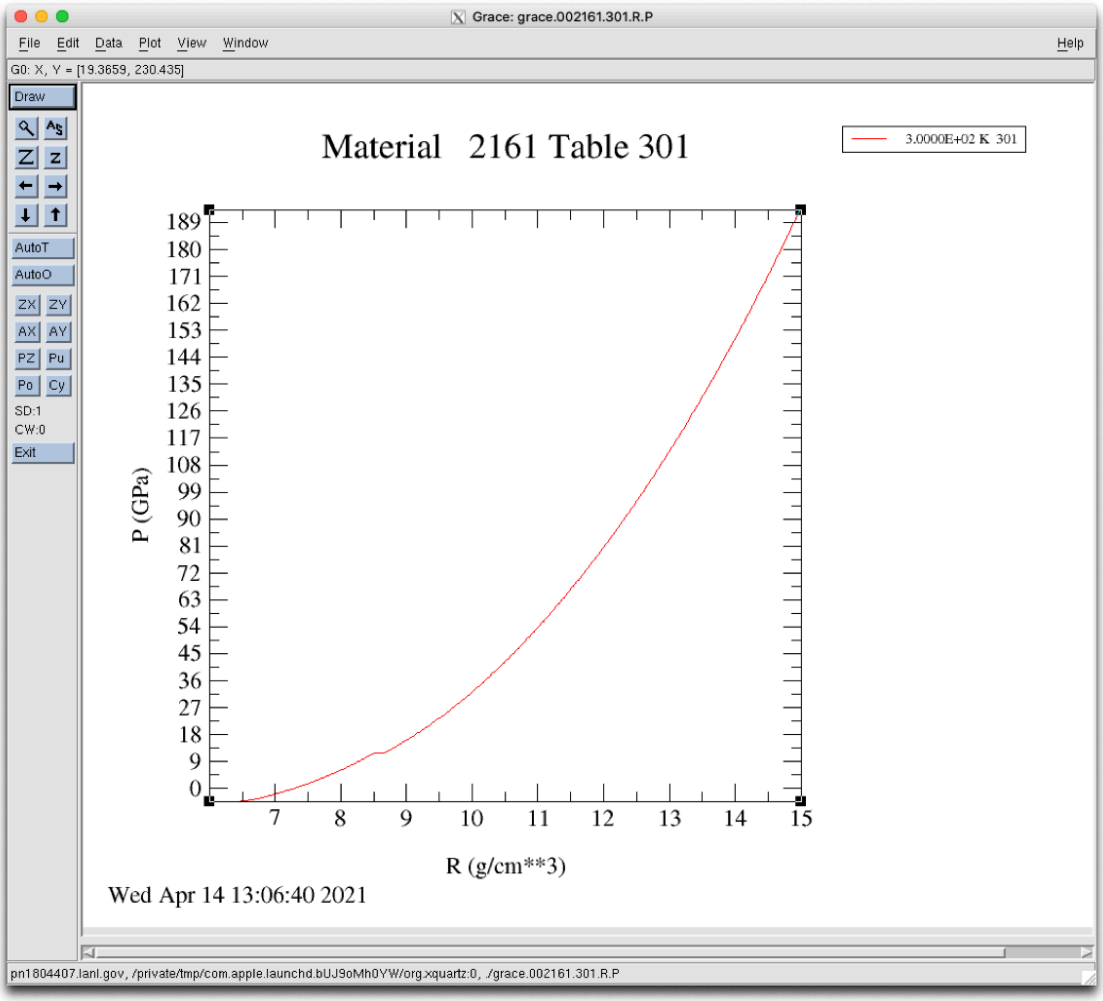
Click Continue



For our purposes, choose linear in x and y



We will plot and then terminate the application


Grace: grace.002161.301.R.P
File Edit Data Plot View Window Help
G0: X, Y = [19.3659, 230.435]
Draw
AutoT
AutoO
ZX ZY
AX AY
PZ Pu
Po Cy
SD:1
CW:0
Exit
Material 2161 Table 301
3.0000E+02 K 301
P (GPa)
189
180
171
162
153
144
135
126
117
108
99
90
81
72
63
54
45
36
27
18
9
0
7 8 9 10 11 12 13 14 15
R (g/cm**3)
Wed Apr 14 13:06:40 2021
pn1804407.lanl.gov, /private/tmp/com.apple.launchd.bUJ9oMh0YW/org.xquartz:0, ./grace.002161.301.R.P

After running the GUI, some files are automatically generated and saved in the `gui` folder. Listing 10 shows the contents of the `gui/sesamein.user` file created by the GUI. The options we chose in the GUI are now stored in this file and we can run opensesame on this file to create the isotherm plot. The difference with this option from using the `xmgrace` file is that if the contents of the 301 table in `lib/002161` have changed, we will get a fresh plot of the new data. This is in contrast to running `xmgrace` on the `grace.002161.301.R.P` file, since that file contains the actual data points for the original isotherm generated when running the GUI the first time.

```
&job job_type='plot' sourcelib_path='../lib' /

&plot
 material_numbers=2161
 curve_type(1)='isotherm'
 curve_material(1)=1
 curve_table_number(1)=301
 curve_x_axis_name(1)='R'
 curve_y_axis_name(1)='P'
 curve_x_lin(1)='lin'
 curve_y_lin(1)='lin'
 curve_number_of_points(1)=0
 curve_grid_lin(1)=.t.
 curve_t0(1)=300
 curve_rho0(1)=-1
 curve_rholo(1)=6
 curve_rhohi(1)=15
 curve_nskip(1)=1
 number_of_curves=1
/
```

Listing 10: `gui/sesamein.user`: auto-generated input file with selected GUI options

The benefit of the `sesamein.user` file is that we do not have to rerun the GUI each time and select all of those options again, but instead can run opensesame directly to get a fresh plot with the updated EOS data:

```
$ opensesame <  sesamein.user
```

If we prefer to use something other than `xmgrace` to plot data, we can modify the sesamein.user file slightly to generate a data file, shown in Listing 11.

```
&job job_type='plot' sourcelib_path='../lib' /

&plot
 output_device = 'data_file'      ! add save to file option
 plot_file_name = 'isotherm.dat' ! name of file to save to
 material_numbers=2161
 curve_type(1)='isotherm'
 curve_material(1)=1
 curve_table_number(1)=301
 curve_x_axis_name(1)='R'
 curve_y_axis_name(1)='P'
 curve_x_lin(1)='lin'
 curve_y_lin(1)='lin'
 curve_number_of_points(1)=0
 curve_grid_lin(1)=.t.
 curve_t0(1)=300
 curve_rho0(1)=-1
 curve_rholo(1)=6
 curve_rhohi(1)=15
 curve_nskip(1)=1
 number_of_curves=1
/
```

Listing 11: `gui/sesamein.user`: updated with options to save to a file

Here we have added these two lines with added comments that will make save the isotherm to a file called `isotherm.dat`.

The file `isotherm.dat` has the form of Listing 12.

```
# R "P  3.0000E+02 K  301"
  0.601201425000000D+01 -0.425908007663237D+01
  0.619419650000000D+01 -0.424562744668274D+01
  0.637637875000000D+01 -0.424452474215052D+01
  0.655856100000000D+01 -0.418344661032679D+01
  0.674074325000000D+01 -0.332385879461467D+01
  0.692292550000000D+01 -0.233533299185693D+01
  0.699579840000000D+01 -0.190578142291123D+01
  0.701562539427208D+01 -0.178551433934686D+01
  0.703545238854415D+01 -0.166376993441048D+01
  ...
```

Listing 12: `gui/isotherm.dat`: isotherm data generated using save data option

Notice that the Fortran convention of using D for exponential format is used. This causes some problems if plotting the data in other languages such as Python. A simple command to change the D to E using `sed` is,

```
$  sed -i .bak 's/D/E/g'  isotherm.dat
```

With the `isotherm.dat` file, we can now use other languages like Python to plot with other data. An example of a Python script that plots the 2161 isotherm with a comparison

to experimental data is shown in Listing 13, which uses matplotlib [16] and numpy [17]. The plot generated from running this script is shown in Fig. 3.

```
from pylab import *
import os
rcParams.update({'font.size':48, 'text.usetex': True})

os.system("sed -i .bak 's/D/E/g' isotherm.dat")

data = genfromtxt('isotherm.dat')       # load isotherm data
exp  = genfromtxt('2013-salamat.dat')   # load experimental data

figure(figsize=(14,12))
plot(data[:,0], data[:,1],lw=3, label='2161 EOS')
plot(exp[:,0], exp[:,1], 'o', label = 'Salamat et al. (2013)')
legend(fontsize=24)
xlim(7,14)
ylim(0,150)
ylabel('$P$ (GPa)')
xlabel(r'$\rho$ (g/cm$^3$)')
tight_layout()
show()
```

Listing 13: `gui/plot.dat`: Python script for plotting an isotherm with experimental data

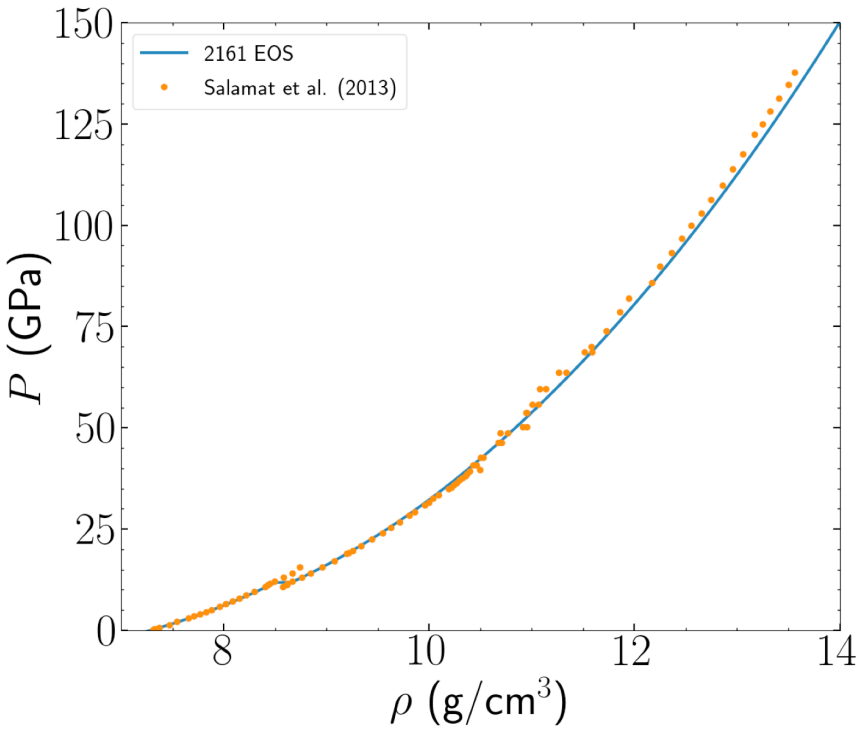


Figure 3: Isotherm of 2161 and data from Salamat et al. (2013) [18].

4. Creating ASCII-formatted EOS tables

Typically, hydro code users use either ASCII-formatted EOS tables as input to hydro codes. To demonstrate how to do this, we create a new folder titled `5-ascii`, so that we have

```
$ ls
1-beta/    2-gamma/    3-liquid/    4-multiphase/    5-ascii    lib/
```

We then create another input file, `5-ascii/input.nml` with the contents shown in Listing 14. Running `opensesame < input.nml` in that folder will generate a file called `ascii-2161` that can be shared with hydro code users.

```
&job job_type = 'makeasciifile'
  sourcelib_path = '../lib/'  ! create ASCII using contents of lib folder
  resultlib_path = '../lib/'
  sesame_ascii_path = './ascii-2161' ! create ASCII file with name ascii-2161
/
```

Listing 14: `5-ascii/input.nml`: file for generating ASCII formatted tables

Note that with the ASCII file, it is possible to generate a binary-formatted version of the ASCII file using EOSPAC. We do not cover that here since it involves the use of EOSPAC.

References

[1] D. George, "Opensesame," *Los Alamos National Laboratory Tech. Rep. No. LA-CC-03-0087*, 2003.

[2] E. Chisolm, C. Greeff, and D. George, "Constructing explicit multiphase equations of state with opensesame," *Los Alamos National Laboratory Tech. Rep. No. LA-UR-05-9413*, 2005.

[3] S. D. Crockett, "Fundamentals of sesame equations of state (eos)," *Los Alamos National Laboratory Tech. Rep. No. LA-UR 08-03679*, 2008.

[4] C. Greeff, E. Chisolm, D. George, C. Greeff, E. Chisolm, and D. George, "SESAME 2161: An explicit multiphase equation of state for tin," *Los Alamos National Laboratory Tech. Rep. No. LA-UR-05-9414*, 2005.

[5] D. A. Rehn, C. W. Greeff, L. Burakovsky, D. G. Sheppard, and S. D. Crockett, "Multiphase tin equation of state using density functional theory," *Physical Review B*, 2021.

[6] D. A. Rehn, C. W. Greeff, D. G. Sheppard, and S. D. Crockett, "Using density functional theory to construct multiphase equations of state: tin as an example," *Los Alamos National Laboratory Tech. Rep. No. LA-UR-20-29170*, 2020.

[7] M. Metcalf, J. Reid, and M. Cohen, *Modern fortran explained*. Oxford University Press, 2011.

[8] P. Vinet, J. R. Smith, J. Ferrante, and J. H. Rose, "Temperature effects on the universal equation of state of solids," *Physical Review B*, vol. 35, no. 4, p. 1945, 1987.

[9] P. Debye, "Zur theorie der spezifischen wärmen," *Annalen der Physik*, vol. 344, no. 14, pp. 789–839.

[10] E. Chisolm, "Thomas–fermi–dirac theory as calculated in the EOS-production programs Grizzly and Opensesame," tech. rep., Report no. LA-UR-05-2297, Los Alamos: Los Alamos National Lab, 2003.

[11] E. Fermi, "Un metodo statistico per la determinazione di alcune priorieta dell'atome," *Rend. Accad. Naz. Lincei*, vol. 6, no. 602-607, p. 32, 1927.

[12] L. H. Thomas, "The calculation of atomic fields," in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 23, pp. 542–548, Cambridge University Press, 1927.

[13] E. D. Chisolm, S. D. Crockett, and D. C. Wallace, "Test of a theoretical equation of state for elemental solids and liquids," *Physical Review B*, vol. 68, no. 10, p. 104103, 2003.

[14] E. Chisolm, S. Crockett, and D. Wallace, "Extending the CCW EOS I: Extending the cold and nuclear contributions to high compression," tech. rep., LA-UR-03-7344, Los Alamos National Laboratory, 2005.

[15] D. George, "Users manual opensesame," *Los Alamos National Laboratory Tech. Rep. No. LA-UR-04-1585*, 2004.

[16] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[17] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[18] A. Salamat, R. Briggs, P. Bouvier, S. Petitgirard, A. Dewaele, M. E. Cutler, F. Cora, D. Daisenberger, G. Garbarino, and P. F. McMillan, "High-pressure structural transformations of sn up to 138 gpa: Angle-dispersive synchrotron x-ray diffraction study," *Physical Review B*, vol. 88, no. 10, p. 104104, 2013.